کامپایلر
پویشگر
مقدمات و خودکاره متناهی (معین) و عبارت منظم

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

خودکاره متناهی و پویشگر

نمودار انتقال

- به مثابه انتزاعی از کد موردنیاز پیاده‌سازی
- یا شی ریاضی با عنوان خودکاره متناهی **finite automata** خم

تعریف خودکاره متناهی

- پنج‌تائی (S,Σ,δ,s_0,F)
  - S مجموعه متناهی از حالت‌های تشخیص‌گر (دارای حالت خطا s_e)
  - Σ مجموعه متناهی از الفباء مورداستفاده تشخیص‌گر
  - δ *تابع انتقال که* به ازای حالت فعلی و ورودی الفبا به حالتی دیگر می‌رود
  - s_0 حالت آغاز
  - F مجموعه حالات پذیرش (نهائی) زیرمجموعه‌ای از S. اگر با خواندن تمامی رشته ورودی، خودکاره در یکی از حالات پذیرش باشد، رشته پذیرفته شده است.

خودکاره متناهی - *ادامه*

مثال- تشخیص not و new و while

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_e\}$$

$$\Sigma = \{\mathtt{e}, \mathtt{h}, \mathtt{i}, \mathtt{l}, \mathtt{n}, \mathtt{o}, \mathtt{t}, \mathtt{w}\}$$

$$\delta = \left\{ \begin{array}{lllll} s_0 \xrightarrow{\mathtt{n}} s_1, & s_0 \xrightarrow{\mathtt{w}} s_6, & s_1 \xrightarrow{\mathtt{e}} s_2, & s_1 \xrightarrow{\mathtt{o}} s_4, & s_2 \xrightarrow{\mathtt{w}} s_3, \\ s_4 \xrightarrow{\mathtt{t}} s_5, & s_6 \xrightarrow{\mathtt{h}} s_7, & s_7 \xrightarrow{\mathtt{i}} s_8, & s_8 \xrightarrow{\mathtt{l}} s_9, & s_9 \xrightarrow{\mathtt{e}} s_{10} \end{array} \right\}$$

$$s_0 = s_0$$

$$\mathrm{F} = \{s_3, s_5, s_{10}\}$$

به ازای تمامی دیگر ترکیب‌های حالات s_i و نویسه ورودی c: $\delta(s_i, c) = s_e$

- s_e حالت خطا

نمودار انتقال «متناظر» خودکاره متناهی

خودکاره متناهی - *ادامه*

پذیرش رشته $\boldsymbol{x}$

- اگر و فقط اگر انتقال بین حالات خواندن رشته‌های $\boldsymbol{x}$ در حالتی از حالات پذیرش تمام شود.
- مثال s_3 برای new

در قالب ریاضی:

اگر $\boldsymbol{x} = x_1x_2x_3 \dots x_n$، آن‌گاه خم (S,Σ,δ,s_0,F)، $\boldsymbol{x}$ را می‌پذیرد اگر و فقط اگر

$$\delta(\delta(\dots\delta(\delta(\delta(s_0,x_1),x_2),x_3)\dots,x_{n-1}),x_n) \in F$$

$\delta(s_0,x_1)$ نخستین انتقال

خودکاره متناهی - *ادامه*

نتایج حاصل از خواندن رشته‌ها

- سه مورد

خودکاره متناهی - *ادامه*

نتایج حاصل از خواندن رشته‌ها

- سه مورد

۱- پذیرش رشته

خودکاره متناهی - *ادامه*

نتایج حاصل از خواندن رشته‌ها

- سه مورد

۱- پذیرش رشته

۲- خطای لغوی رفتن به حالت خطا

- رشتهٔ ورودی «پیشوند» معتبر هیچ لغتی نیست

خودکاره متناهی - *ادامه*

نتایج حاصل از خواندن رشته‌ها

- سه مورد

۱- پذیرش رشته

۲- خطای لغوی رفتن به حالت خطا

- رشتهٔ ورودی «پیشوند» معتبر هیچ لغتی نیست

۳- ماندن در حالت غیرپایانی

پیشوند و پسوند و زیررشته و زیردنباله

پیشوند **prefix**: هر رشتهٔ حاصل از حذف هیچ یا بیشتر از نشان‌های انتهای رشته اصلی

پسوند **suffix**: هر رشتهٔ حاصل از حذف هیچ یا بیشتر از نشان‌های ابتدای رشته اصلی

زیررشته **substring**: رشتهٔ حاصل از حذف پیشوند یا پسوند رشتهٔ اصلی

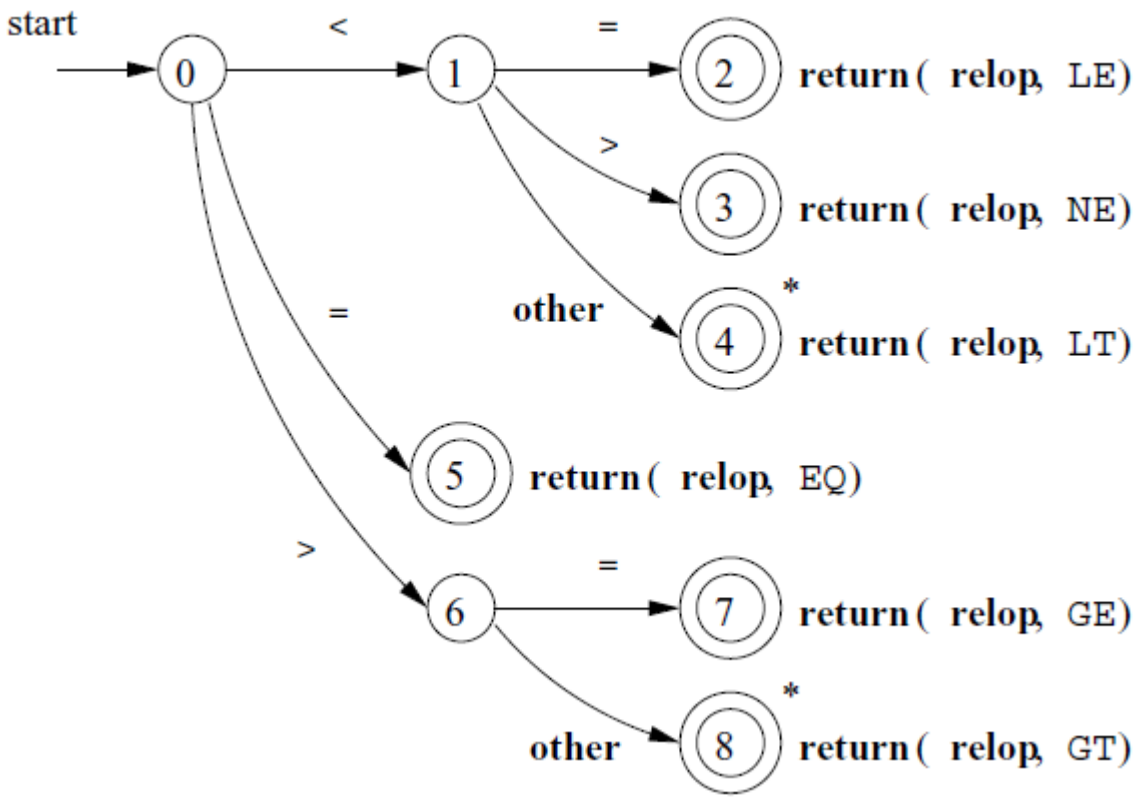
زیردنباله **subsequence**: رشتهٔ حاصل از حذف چند نویسه که لزوما دنبال هم نباشند

پیشوند و پسوند و زیررشتهٔ محض (سره)

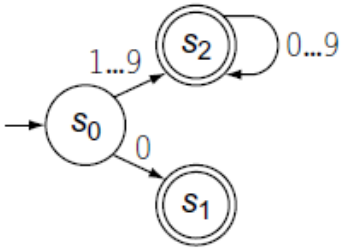
مثال – تکه‌ها و الگوهای متناظر و مقادیر

Lexemes	Token Name	Attribute Value
Any *ws*	–	–
`if`	**if**	–
`then`	**then**	–
`else`	**else**	–
Any *id*	**id**	Pointer to table entry
Any *number*	**number**	Pointer to table entry
`<`	**relop**	`LT`
`<=`	**relop**	`LE`
`=`	**relop**	`EQ`
`<>`	**relop**	`NE`
`>`	**relop**	`GT`
`>=`	**relop**	`GE`

نمودار انتقال مقایسه‌گرها



تشخیص کلمات پیچیده‌تر



تشخیص عدد

- عدد صحیح بی‌علامت نامنفی

```
char ← NextChar( );
state ← s0 ;

while (char ≠ eof and state ≠ se) do
   state ← δ(state,char);
   char ← NextChar( );
end;

if (state ∈ SA)
    then report acceptance;
    else report failure;
```

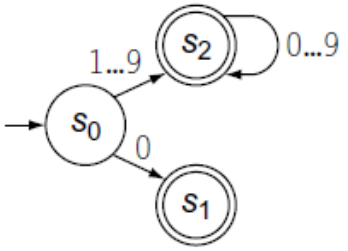
$$S = \{s_0, s_1, s_2, s_e\}$$

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$\delta = \left\{ \begin{matrix} s_0 \xrightarrow{0} s_1, & s_0 \xrightarrow{1\text{-}9} s_2 \\ s_2 \xrightarrow{0\text{-}9} s_2, & s_1 \xrightarrow{0\text{-}9} s_e \end{matrix} \right\}$$

$$S_A = \{s_1, s_2\}$$

تشخیص کلمات پیچیده‌تر - *ادامه*



تشخیص عدد

- عدد صحیح بی‌علامت نامنفی

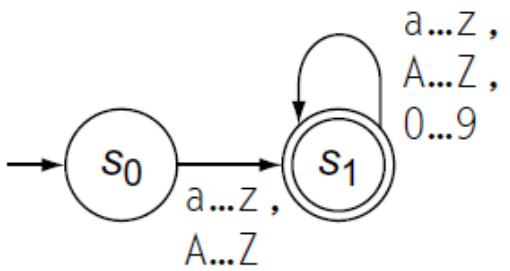
δ	0	1	2	3	4	5	6	7	8	9	Other
s_0	s_1	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_e
s_1	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e
s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_e
s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e

تشخیص کلمات پیچیده‌تر - *ادامه*

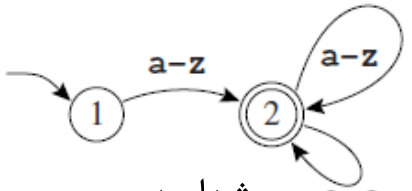
تشخیص شناسه‌ها

تشخیص کلمات پیچیده‌تر - *ادامه*

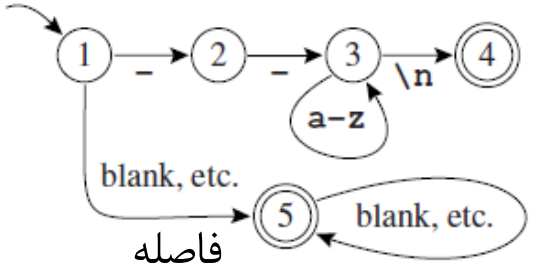
تشخیص شناسه‌ها



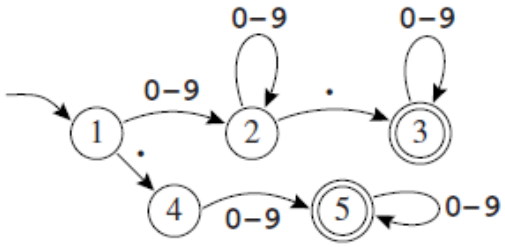
IF



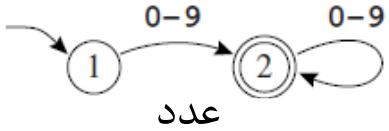
شناسه



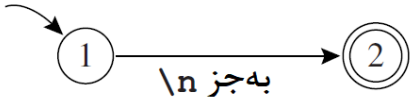
فاصله



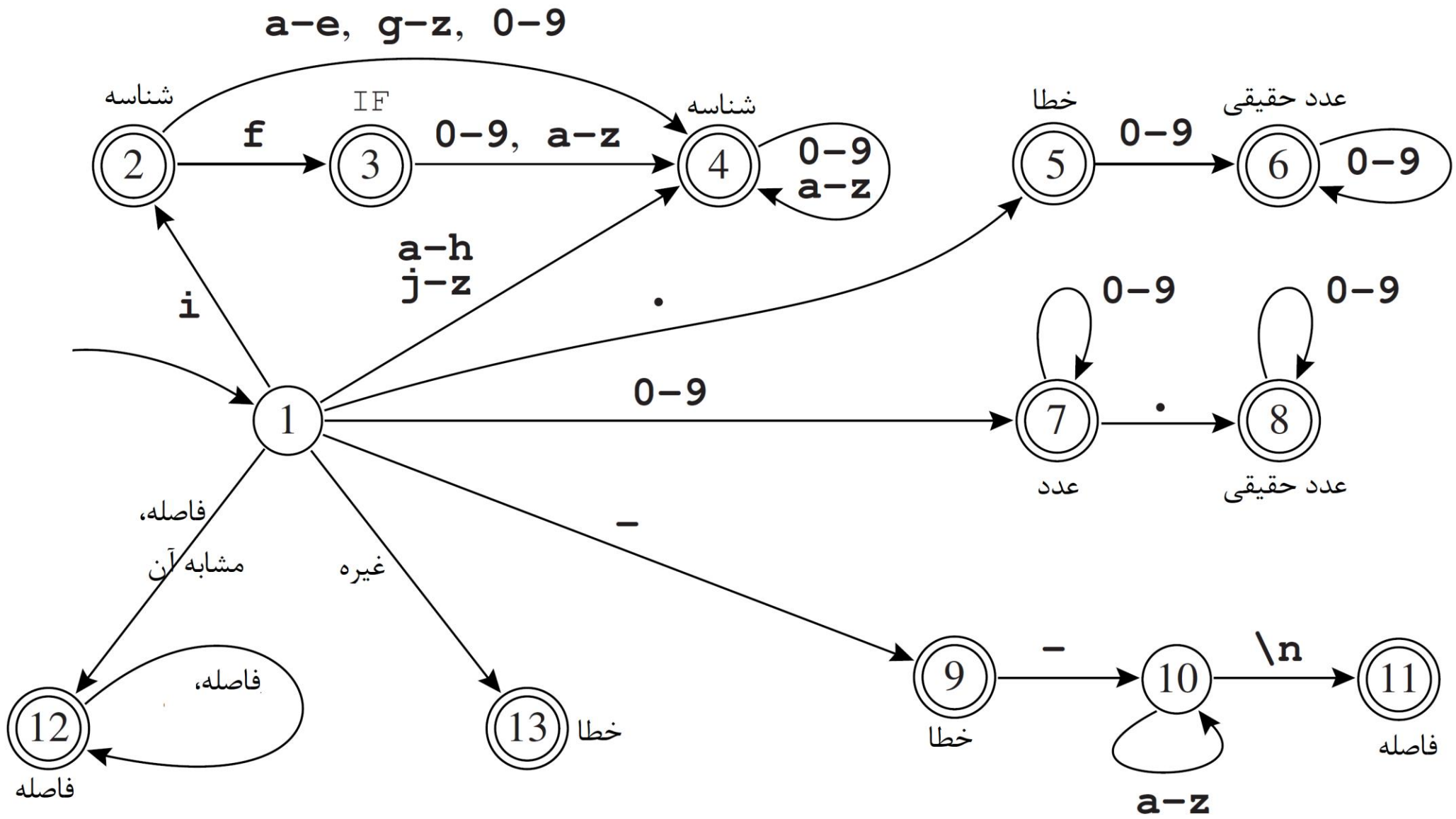
عدد حقیقی



عدد



خطا


a-e, g-z, 0-9
شناسه
IF
شناسه
f
0-9, a-z
0-9
a-z
خطا
عدد حقیقی
0-9
0-9
a-h
j-z
i
.
0-9
0-9
0-9
0-9
.
عدد
عدد حقیقی
فاصله،
مشابه آن
غیره
_
فاصله،
-
\n
a-z
فاصله
خطا
خطا
فاصله
1
2
3
4
5
6
7
8
9
10
11
12
13

امکان نمایش با ماتریس انتقال

```
int edges[][256]={ /* ···0 1 2···-···e f g h i j··· */
/* state 0 */     {0,0,···0,0,0···0···0,0,0,0,0,0···},
/* state 1 */     {0,0,···7,7,7···9···4,4,4,4,2,4···},
/* state 2 */     {0,0,···4,4,4···0···4,3,4,4,4,4···},
/* state 3 */     {0,0,···4,4,4···0···4,4,4,4,4,4···},
/* state 4 */     {0,0,···4,4,4···0···4,4,4,4,4,4···},
/* state 5 */     {0,0,···6,6,6···0···0,0,0,0,0,0···},
/* state 6 */     {0,0,···6,6,6···0···0,0,0,0,0,0···},
/* state 7 */     {0,0,···7,7,7···0···0,0,0,0,0,0···},
/* state 8 */     {0,0,···8,8,8···0···0,0,0,0,0,0···},
    et cetera
}
```

امکان نمایش با ماتریس انتقال - *ادامه*

ردیف‌ها حالت‌ها

ستون‌ها حرف ورودی

حالت صفر

- **حالت مرده**
- **طوقه به ازای تمامی ورودی‌ها**
- **جهت تدوین عدم وجود یال**

تشخیص طولانی‌ترین انطباق

طولانی‌ترین زیررشته ورودی که معتبر است

نیاز به حفظ رهگیری طولانی‌ترین انطباقی دیده شده و مکان انطباق

؟

تشخیص طولانی‌ترین انطباق

طولانی‌ترین زیررشته ورودی که معتبر است

نیاز به حفظ رهگیری طولانی‌ترین انطباقی دیده شده و مکان انطباق

؟

- به حافظه سپردن آخرین باری که خودکاره در حالت پذیرش بوده است

تشخیص طولانی‌ترین انطباق

طولانی‌ترین زیررشته ورودی که معتبر است

نیاز به حفظ رهگیری طولانی‌ترین انطباقی دیده شده و مکان انطباق

؟

- به حافظه سپردن آخرین باری که خودکاره در حالت پذیرش بوده است
- چگونه؟

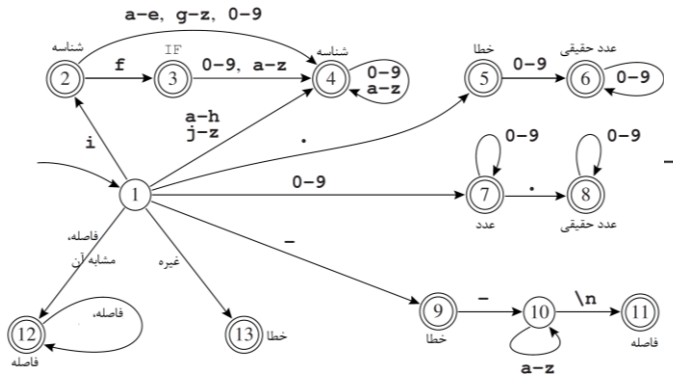
تشخیص طولانی‌ترین انطباق

طولانی‌ترین زیررشته ورودی که معتبر است

نیاز به حفظ رهگیری طولانی‌ترین انطباقی دیده شده و مکان انطباق

؟

- به حافظه سپردن آخرین باری که خودکاره در حالت پذیرش بوده است
- چگونه؟ با استفاده از دو متغیر
- متغیر «آخرین-پذیرش»
  - اخرین حالت پذیرش دیده‌شده
- متغیر «محل‌شروع-آخرین-پذیرش»



پذیرش؟	ورودی فعلی	حالت فعلی	آخرین پذیرش
	⊤if --not-a-com	1	0
	|i⊤f --not-a-com	2	2
	|if⊤ --not-a-com	3	3
return IF	|if⊤⊥--not-a-com	0	3
	if⊤ --not-a-com	1	0
	if|⊤--not-a-com	12	12
found white space; resume	if|⊤⊥-not-a-com	0	12
	if ⊤--not-a-com	1	0
	if |⊤-not-a-com	9	9
	if |-⊤not-a-com	10	9
	if |-⊤not-a-com	10	9
	if |-⊤not-a-com	10	9
	if |-⊤not-a-com	10	9
error, illegal token '-'; resume	if |-⊤not-a-com	0	9
	if -⊤-not-a-com	1	0
	if -|⊤not-a-com	9	9
error, illegal token '-'; resume	if -|⊤not-a-com	0	9

|	حرف فعلی
⊥	محلی فعلی خودکاره
⊤	محل آخرین حرفی که خودکاره در حالت پذیرش بوده است

تشخیص کلمات پیچیده‌تر - *ادامه*

خم به مثابه مشخصات مورداستفاده تشخیص‌گر

اما جمع و جور نیست

طراحی سخت

؟

عبارت منظم

راه‌حل: طراحی با عبارت منظم (ع‌م)

متناظر خودکاره متناهی

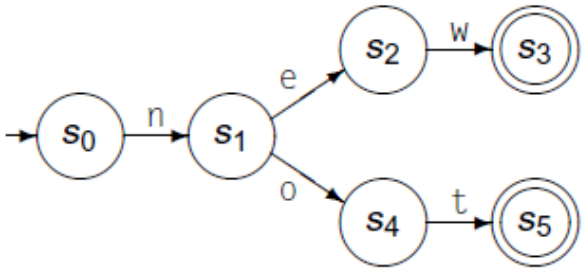
- اثبات در درس نظریه زبان‌ها و خودکاره‌ها

عبارت منظم

راه‌حل: طراحی با عبارت منظم (ع‌م)

متناظر خودکاره متناهی

- اثبات در درس نظریه زبان‌ها و خودکاره‌ها



عبارت منظم

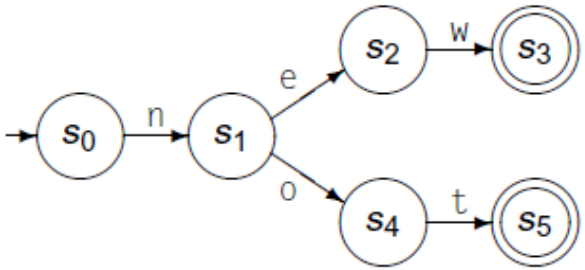
راه‌حل: طراحی با عبارت منظم (ع‌م)

متناظر خودکاره متناهی

- اثبات در درس نظریه زبان‌ها و خودکاره‌ها

new | while

n(ew| ot)



عبارت منظم - تعریف صوری

۱- نویسه

۲- رشته تهی

۳- اگر r و s عبارت منظم باشند، آن‌گاه

- اتصال آنها rs
- اجتماع آنها r | s
- بستار کلین آن نیز r*

- اتصال و اجتماع و بسط سه عمل پایه‌ای ایجاد عبارات منظم

عبارت منظم - مثال

اعداد دودوئی مضرب ۲

عبارت منظم - مثال

اعداد دودوئی مضرب ۲

$$(0|1)^*0$$

عبارت منظم - مثال

اعداد دودوئی مضرب ۲

$$(0|1)^*0$$

رشته‌هائی از a و b بدون دو a متوالی

عبارت منظم - مثال

اعداد دودوئی مضرب ۲

$$(0|1)^*.0$$

رشته‌هائی از a و b بدون دو a متوالی

$$b^*(abb^*)^*(a|\epsilon)$$

عبارت منظم - مثال

اعداد دودوئی مضرب ۲

$$(0|1)^*.0$$

رشته‌هائی از a و b بدون دو a متوالی

$$b^*(abb^*)^*(a|\epsilon)$$

رشته‌هائی از a و b شامل دو a متوالی

عبارت منظم - مثال

اعداد دودوئی مضرب ۲

$$(0|1)^*.0$$

رشته‌هائی از a و b بدون دو a متوالی

$$b^*(abb^*)^*(a|\epsilon)$$

رشته‌هائی از a و b شامل دو a متوالی

$$(a|b)^*aa(a|b)^*$$

عبارت منظم – ویژگی‌ها

عملیات	تعریف
اجتماع دو زبان	$L \cup M = \{s \mid s \in L \text{ یا } s \in M\}$
اتصال دو زبان	$LM = \{st \mid s \in L, t \in M\}$
بستار کلین زبان	$L^* = \bigcup_{i=0}^{\infty} L^i$
بستار مثبت زبان	$L^+ = \bigcup_{i=1}^{\infty} L^i$
یک یا هیچ نمونه	$r? = r \mid \epsilon$
رده‌های نویسه	$a_1 \mid a_2 \mid \ldots \mid a_n = [a_1 a_2 \ldots a_n]; [a_1 - a_n]$

عبارت منظم – ویژگی‌ها

بسته بودن نسبت به

- اتصال
- اجتماع
- بستار کلین

- ترتیب اولویت عملیات‌ها زیاد به کم (همگی شرکت‌پذیر از چپ)
  - *
  - اتصال
  - یا

عبارت منظم – مثال ۲

شناسه در زبان‌های الگول-محور: $([A-Z] \mid [a-z])\ ([A-Z] \mid [a-z] \mid [0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0 \mid [1-9][0-9]^*$$

عبارت منظم - مثال۲

شناسه در زبان‌های الگول-محور: $([A-Z] \mid [a-z])\ ([A-Z] \mid [a-z] \mid [0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0 \mid [1-9][0-9]^*$$

عدد صحیح با علامت (0125 قبول!)

عبارت منظم – مثال ۲

شناسه در زبان‌های الگول-محور: $([A-Z] \mid [a-z])\ ([A-Z] \mid [a-z] \mid [0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0 \mid [1-9][0-9]^*$$

عدد صحیح با علامت (0125 قبول!)

$$[+ \mid - \mid \epsilon][0-9]^+$$

عبارت منظم – مثال ۲

شناسه در زبان‌های الگول-محور: $([A-Z] \mid [a-z])\ ([A-Z] \mid [a-z] \mid [0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0 \mid [1-9][0-9]^*$$

عدد صحیح با علامت (0125 قبول!)

$$[+\mid -\mid \epsilon][0-9]^+$$
$$[+\mid -]?\,[0-9]^+$$

عبارت منظم – مثال ۲

شناسه در زبان‌های الگول-محور: $([A-Z] \mid [a-z])\ ([A-Z] \mid [a-z] \mid [0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0 \mid [1-9][0-9]^*$$

عدد صحیح با علامت (0125 قبول!)

$$[+\mid -\mid \epsilon][0-9]^+$$
$$[+\mid -]?\,[0-9]^+$$

عدد ممیز شناور

عبارت منظم – مثال ۲

شناسه در زبان‌های الگول-محور: $([A-Z]\mid[a-z])\ ([A-Z]\mid[a-z]\mid[0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0\mid[1-9][0-9]^*$$

عدد صحیح با علامت (0125 قبول!)

$$[+\mid-\mid\epsilon][0-9]^+$$
$$[+\mid-]?\,[0-9]^+$$

عدد ممیز شناور

$$[+\mid-\mid\epsilon]([0-9]^+\mid([0-9]^*.\,[0-9]^+))$$

عبارت منظم – مثال ۲

شناسه در زبان‌های الگول-محور: $([A-Z]\mid[a-z])\ ([A-Z]\mid[a-z]\mid[0-9])^*$

- بعضی دیگر شامل ـ و & و %
- محدود بودن طول

عدد صحیح بدون علامت

$$0\mid[1-9][0-9]^*$$

عدد صحیح با علامت (0125 قبول!)

$$[+\mid-\mid\epsilon][0-9]^+$$
$$[+\mid-]?\,[0-9]^+$$

عدد ممیز شناور

$$[+\mid-\mid\epsilon]([0-9]^+\mid([0-9]^*.\,[0-9]^+))$$
$$([+\mid-]?\,[0-9]^+.?\,[0-9]^+)\mid([+\mid-]?\,[0-9]^+[.\,[0-9]^+]?)$$

عبارت منظم – مثال۳

d-ها عضوی از متغیرها

r-ها زیرمجموعه‌ای از حروف و متغیرها

$$
\begin{aligned}
d_1 &\rightarrow r_1 \\
d_2 &\rightarrow r_2 \\
&\cdots \\
d_n &\rightarrow r_n
\end{aligned}
$$

عبارت منظم – مثال۳

شناسه‌های زبان سی

- رشته‌هائی مرکب از حروف و ارقام و خط‌تیره

$$
\begin{aligned}
letter\_ &\rightarrow \texttt{A} \mid \texttt{B} \mid \cdots \mid \texttt{Z} \mid \texttt{a} \mid \texttt{b} \mid \cdots \mid \texttt{z} \mid \texttt{\_} \\
digit &\rightarrow \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{9} \\
id &\rightarrow letter\_ \; (\; letter\_ \mid digit \;)^*
\end{aligned}
$$

- اعداد بی‌علامت

$$
\begin{aligned}
digit &\rightarrow \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{9} \\
digits &\rightarrow digit \; digit^* \\
optionalFraction &\rightarrow \texttt{.} \; digits \mid \epsilon \\
optionalExponent &\rightarrow (\; \texttt{E} \; (\; \texttt{+} \mid \texttt{-} \mid \epsilon \;) \; digits \;) \mid \epsilon \\
number &\rightarrow digits \; optionalFraction \; optionalExponent
\end{aligned}
$$

عبارت منظم – مثال۳

شناسه‌های زبان سی

- رشته‌هائی مرکب از حروف و ارقام و خط‌تیره

$$
\begin{aligned}
letter\_ &\rightarrow [\texttt{A-Za-z\_}] \\
digit &\rightarrow [\texttt{0-9}] \\
id &\rightarrow letter\_\ (\ letter\_ \mid digit\ )^*
\end{aligned}
$$

- اعداد بی‌علامت

$$
\begin{aligned}
digit &\rightarrow [\texttt{0-9}] \\
digits &\rightarrow digit^+ \\
number &\rightarrow digits\ (\texttt{.}\ digits)?\ (\ \texttt{E}\ [\texttt{+-}]?\ digits\ )?
\end{aligned}
$$

عبارت منظم – مثال۴

شبیه مثال قبل با تفسیر دیگر

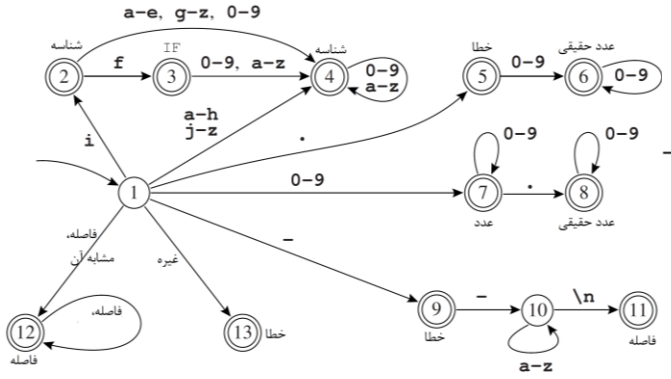
- نحو اعداد ورودی ماشین حساب دستی

$$
\begin{aligned}
number &\longrightarrow integer \mid real \\
integer &\longrightarrow digit\ digit^* \\
real &\longrightarrow integer\ exponent \mid decimal\ (exponent \mid \epsilon) \\
decimal &\longrightarrow digit^*\ (\ .\ digit \mid digit\ .\ )\ digit^* \\
exponent &\longrightarrow (\ \mathtt{e} \mid \mathtt{E}\ )\ (\ \mathtt{+} \mid \mathtt{-} \mid \epsilon\ )\ integer \\
digit &\longrightarrow \mathtt{0} \mid \mathtt{1} \mid \mathtt{2} \mid \mathtt{3} \mid \mathtt{4} \mid \mathtt{5} \mid \mathtt{6} \mid \mathtt{7} \mid \mathtt{8} \mid \mathtt{9}
\end{aligned}
$$

- هیچ‌چیز بر اساس خودش تعریف نشده است
  - تعریف بازگشتی صفت مشخصه و متمایزکننده دستور مستقل‌ازمتن

خواص جبری حاکم بر عبارت منظم

ویژگی	
$r\|s = s\|r$	جابجایی
$r\|(s\|t) = (r\|s)\|t$	انجمنی
$r(st) = (rs)t$	انجمنی اتصال
$r(s\|t) = rs\|rt; (s\|t)r = sr\|tr$	توزیع‌پذیری اتصال
$\epsilon r = r\epsilon = r$	
$r^* = (r\|\epsilon)^*$	
$r^{**} = r^*$	بی‌اثری



پذیرش؟	ورودی فعلی	حالت فعلی	آخرین پذیرش
	⊤i⊥f --not-a-com	1	0
	|i⊤f⊥ --not-a-com	2	2
	|if⊤ --not-a-com	3	3
return IF	|if⊤⊥--not-a-com	0	3
	if⊤ --not-a-com	1	0
	if|⊤--not-a-com	12	12
found white space; resume	if|⊤⊥-not-a-com	0	12
	if ⊤--not-a-com	1	0
	if |⊤-not-a-com	9	9
	if |-⊤not-a-com	10	9
	if |-⊤not-a-com	10	9
	if |-⊤not-a-com	10	9
	if |-⊤not-a-com	10	9
error, illegal token '-'; resume	if |-⊤not-a-com	0	9
	if -⊤-not-a-com	1	0
	if -|⊤not-a-com	9	9
error, illegal token '-'; resume	if -|⊤not-a-com	0	9

|	حرف فعلی
⊥	محلی فعلی خودکاره
⊤	محل آخرین حرفی که خودکاره در حالت پذیرش بوده است

چند ع‌م-نویسی

```
if                                              {return IF;}
[a-z][a-z0-9]*                                  {return ID;}
[0-9]+                                          {return NUM;}
([0-9]+"."[0-9]*)|([0-9]*"."[0-9]+)             {return REAL;}
("--"[a-z]*"\n")|(" "|"\n"|"\t")+               {  /* do nothing */ }
.                                               {error();}
```

ابهام در انتخاب قواعد

طولانی‌ترین انطباق

- «طولانی‌ترین زیررشتهٔ» ممکن ورودی در انطباق با عبارتی منظم
- انتخاب به عنوان تکهٔ بعدی

قاعده اولویت

- اولویت در عبارت منظم نمایشگر طولانی‌ترین پیشوند
- اهمیت دار بودن ترتیب نوشتن عبارت منظم

زبان طبیعی دربرابر زبان برنامه‌نویسی

تحلیل لغوی محل افتراق زبان طبیعی و زبان برنامه‌نویسی

عدم تطابق بین نمایش لغت (اجزای لغت) و معنای آن در زبان طبیعی

- بو و بود

تمامی ترکیب‌ها درست نیستند.

- بوج و بود

پویشگر زبان طبیعی می‌تواند فنون خم-محور را به کار ببرد ولی کافی نیست

- نیاز به لغت‌نامه جهت تایید درستی لغت

بعضی کلمات دارای چند کاربردند.

- زیبا هم نام است و هم صفت.
- بررسی همسایه‌ها و معنا
- در زبان برنامه‌نویسی لغت بدون بررسی دستوری و معنایی کاملا مشخص
- حتی کلیدواژه‌ها

قالب و الگوی مشخص موجب ساده‌تر شدن کار لغت‌کاو و تجزیه‌گر در ازای کاهیدن سلاست زبان

بعضی از زبان‌های جدید اجازه به استفاده چندکاره به کلمات داده‌اند

- PL/I
  - «کلمه محفوظ» ندارد
- انعطاف بیشتر زبان در مقابل پیچیدگی

زبان طبیعی دربرابر زبان برنامه‌نویسی

تحلیل لغوی محل افتراق زبان طبیعی و زبان برنامه‌نویسی

عدم تطابق بین نمایش لغت (اجزای لغت) و معنای آن در زبان طبیعی

- بو و بود

تمامی ترکیب‌ها درست نیستند.

- بوج و بود

پویشگر زبان طبیعی می‌تواند فنون خم-محور را به کار ببرد ولی کافی نیست

- نیاز به لغت‌نامه جهت تایید درستی لغت

بعضی کلمات دارای چند کاربردند.

- زیبا هم نام است و هم صفت.
- بررسی همسایه‌ها و معنا
- در زبان برنامه‌نویسی لغت بدون بررسی دستوری و معنایی کاملا مشخص
- حتی کلیدواژه‌ها

قالب و الگوی مشخص موجب ساده‌تر شدن کار لغت‌کاو و تجزیه‌گر در ازای کاهیدن سلاست زبان

بعضی از زبان‌های جدید اجازه به استفاده چندکاره به کلمات داده‌اند

- PL/I
  - «کلمه محفوظ» ندارد
  - کلیدواژه دارد!
- انعطاف بیشتر زبان در مقابل پیچیدگی

زبان طبیعی دربرابر زبان برنامه‌نویسی

تحلیل لغوی محل افتراق زبان طبیعی و زبان برنامه‌نویسی

عدم تطابق بین نمایش لغت (اجزای لغت) و معنای آن در زبان طبیعی
- بو و بود

تمامی ترکیب‌ها درست نیستند.
- بوج و بود

پویشگر زبان طبیعی می‌تواند فنون خم-محور را به کار ببرد ولی کافی نیست
- نیاز به لغت‌نامه جهت تایید درستی لغت

بعضی کلمات دارای چند کاربردند.
- زیبا هم نام است و هم صفت.
- بررسی همسایه‌ها و معنا
- در زبان برنامه‌نویسی لغت بدون بررسی دستوری و معنایی کاملا مشخص
- حتی کلیدواژه‌ها

قالب و الگوی مشخص موجب ساده‌تر شدن کار لغت‌کاو و تجزیه‌گر در ازای کاهیدن سلاست زبان

بعضی از زبان‌های جدید اجازه به استفاده چندکاره به کلمات داده‌اند
- PL/I
  - «کلمه محفوظ» ندارد
  - کلیدواژه دارد!
- انعطاف بیشتر زبان در مقابل پیچیدگی

```
IF IF = EQUALS THEN
      THEN = ELSE;
ELSE ELSE = IF;
```

منابع

[اژدرها]

[ببر سبز]

[کوپر]

[فیشر]

[نیسان]